View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Research Papers in Economics

A Note on a Rapid Grid Search Method for Solving Dynamic Programming Problems in Economics*

Hui He
Department of Economics
University of Hawai'i at Manoa

and

Hao Zhang
Department of Economics
University of Hawai'i at Manoa

Working Paper No. 10-17
September 17, 2010

Abstract

We introduce a rapid grid search method in solving the dynamic programming problems in economics. Compared to mainstream grid search methods, by using local information of the Bellman equation, this method can significantly increase the efficiency in solving dynamic programming problems by reducing the grid points searched in the control space.

Keywords: Dynamic Programming, Grid Search, Control Space

JEL codes: C63, C61, C68

* We would like to thank Toni Braun, Zhigang Feng, Ayse Imrohoroglu, Selo Imrohoroglu and Thomas Ramsey for their helpful comments.

1 Introduction

High-dimensional dynamic programming (DP) problems have been gaining more and more popularity in economics, yet the solving of high-dimensional DP problems is still quite challenging. For instance, those powerful numerical methods for solving one-dimensional optimization problems, such as golden section search and Brent's method, are difficult to implement in a high-dimensional DP context. On the other hand, grid search, as a widely used numerical method in solving optimization problems, can serve as a stable and reliable method to find solutions to high-dimensional DP problems. Compared to other sophisticated methods, such as Newton's method or the Quasi-Newton method, the basic "brute force" grid search method does not rely on any local or global information of the objective function. In particular, for problems with non-smooth objective functions or multiple local optima, grid search can achieve the global optimum with stable precision and search speed, which methods based on the gradient of objective functions cannot offer. The stability and convergence properties of grid search can be greatly appreciated in the study of high-dimensional DP problems. The drawback of this method, however, is that it can be extremely slow and can impose a huge computational burden in practice. For high precision solutions, the computational cost will increase exponentially, since precision is determined by the fineness of pre-set grid points. The overwhelm-

ing computational cost often interacts with the "curse of dimensionality" arising in high-dimensional DP problems and makes the task of solving these problems using grid search intractable. Therefore, speeding up grid search is a challenging job for computational economists.

Efforts have been made to reduce the computational burden imposed by grid search in DP problems,which can be categorized into two types: reducing grid points searched in the control space or in the state space. İmrohoroğlu, İmrohoroğlu and Joines (1993) make use of the first possibility by applying a bracketing grid search algorithm to solve a dynamic general equilibrium model with imcomplete markets. They first discretize the state and control space by evenly distributed grid points. Starting from coarse grid points to determine an initial optimum, they then make subsequent searches over successively finer grids around the previous optimum. Their method obtains a large improvement in the search speed by reducing grid points searched in the control space.[1] On the other hand, Grune and Semmler (2004) introduce an adaptive grid scheme for DP problems based on local error estimates. Their method reduces the number of grid points searched in the state space and gains great efficiency in computing the models with kinks or steep curvature of the

[1] As shown in Table 1, the total number of grid points that need to be searched by the "brute force" grid search method in the code implemented by İmrohoroğlu, İmrohoroğlu and Joines (1995, 1999) is 1.28073e+10 for 4097 grid points in the state and control space. Using their bracketing method, the actual number of grid points searched is just 1.48898e+8.

value function.

In this paper, we propose a rapid grid search (RGS) method that can significantly enhance the efficiency of solving dynamic problems by reducing the grid points searched in the control space. The idea is to use some local information of the objective function to speed up the searching process. This method has the following advantages. First, it inherits all of the advantages of the mainstream grid search method, such as stability and convergence properties. Second, it requires less information than typical bracketing techniques, such as the algorithm applied by İmrohoroğlu, İmrohoroğlu and Joines (1993). Third, it can be easily embedded in different grid-search-based methods. Therefore, improvement in searching efficiency can be gained without sacrificing the merits of different methods. Last but not least, it can be straightforwardly extended to high-dimensional DP problems with a stable efficiency gain. Although this method is not designed to break the "curse of dimensionality," it helps to reduce considerably the computational cost arising from high-dimensional DP problems.

The remainder of the paper is organized as follows. Section 2 describes the idea of the rapid grid search method and provides the algorithm. Section 3 applies the method to a one-dimensional and a two-dimensional DP example, respectively. Section 4 concludes the paper.

2 Rapid Grid Search Method

In this section we describe a standard Bellman equation arising from a typical DP problem. We then prove a proposition regarding a strictly concave function that is used to build up the intuition of the RGS method. An illustrative algorithm for both one-dimensional and two-dimensional DP problems is also provided.

2.1 Bellman Equation

In a DP model, let $A \in R^L$ be the space of state variables, and let $C \in R^C$ be the space of control variables. Under some conditions, it is well known that the solution of DP problems is equivalent to the following Bellman equation:

$$V_t(a_t) = \max_{c_t \in C} \{U_t(a_t, c_t) + \beta V_{t+1}(a_{t+1})\} \tag{1}$$

subject to

$$\begin{aligned} a_{t+1} &= G_t(a_t, c_t) \\ a_t &\in A \end{aligned}$$

We assume function U_t is strictly concave, function G_t is convex, A and C are compact, and $\beta \in (0,1)$. Given these assumptions, a unique policy function $c_t(a_t)$ of this maximization problem exists. And one can prove that the value function V_t is strictly concave.[2]

In most cases, numerical methods are needed to solve equation (1). For example, the standard grid search method first discretizes the state space A and the control space C by grid points. Then the right-handside of the Bellman equation (1) is evaluated at each grid point and the value is compared until we find the optimum.

2.2 A Proposition of Strictly Concave Function

For a strictly concave function F, we have the following proposition:

Proposition 1 *Let function $f : X \to R$ be strictly concave on a closed set $X \subset R^L$. For x_1, $x_2 \in X$ and $a > 0$, if $f(x_1) > f(x_2)$, we have $f(x_1) > f(x_2 + a(x_2 - x_1))$ and $f(x_2) > f(x_2 + a(x_2 - x_1))$.*

Proof. Since function $f : X \to R$ is strictly concave on closed set X, x_1, $x_2 \in X$ and $a > 0$, we have $x_2 + a(x_2 - x_1) \in X$ and constant $b = \frac{a}{1+a} \in (0,1)$. By Jensen's inequality, we have $bf(x_1)+(1-b)f(x_2+a(x_2-x_1)) < f(bx_1+(1-b)(x_2+a(x_2-x_1))) = f(x_2)$. Hence, $(1-b)f(x_2+a(x_2-x_1)) < f(x_2)-bf(x_1)$. Since $f(x_1) > f(x_2)$, we also

[2] See Stokey, Lucas, and Prescott (1989), Chapter 3, for details.

have $f(x_1)-bf(x_1) > (1-b)f(x_2+a(x_2-x_1))$. Therefore, $f(x_1) > f(x_2+a(x_2-x_1))$. And since $f(x_2) - bf(x_2) > f(x_2) - bf(x_1) > (1-b)f(x_2 + a(x_2 - x_1))$, we also have $f(x_2) > f(x_2 + a(x_2 - x_1))$. ■

This proposition can be interpreted as a binomial relationship between two grid points. Graphically, if the value of one point is higher than another, the value of the higher one dominates the value of any point in X, which is on the extended line along the descending direction of the function value. Figure 1 shows this relationship for a one-dimensional function $f(x)$ for $x \in R^+$. Pick three grid points on the x-axis: x_1, x_2, and x_3. Since $f(x_1) > f(x_2)$, $f(x_1)$ is also higher than $f(x)$ where $x \in [x_2, \infty)$. The optimum must lie in the range between zero and x_2.

Figure 2 shows the intuition of the proposition for a two-dimensional function f. If $f(x_1, y_1) > f(x_2, y_2)$, then for any point on the extended line along the direction from point (x_1, y_1) to point (x_2, y_2), for example, point $(x_3, y_3) = (x_2+\alpha(x_2-x_1), y_2+\alpha(y_2 - y_1))$, $\forall\alpha \in [0, +\infty)$, we have $f(x_1, y_1) > f(x_3, y_3)$. Therefore, we do not need to search any (x_3, y_3).

In a typical DP problem in economics, the utility function U_t and constraint G_t are usually well-defined based on assumptions about preferences and production set. Given the strict concavity of utility function U_t, when we choose c_t over the control space C to maximize the Bellman equation, Proposition 1 can help to dramatically

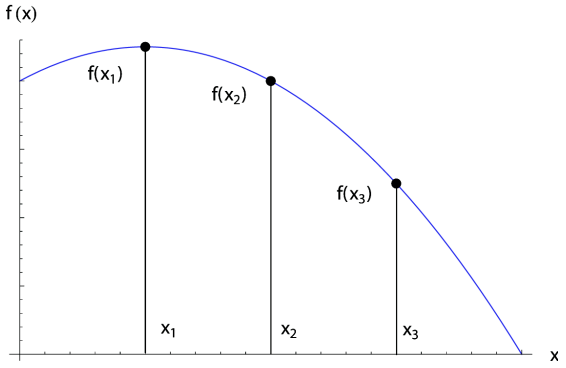

Figure 1: Domination in a one-dimensional case

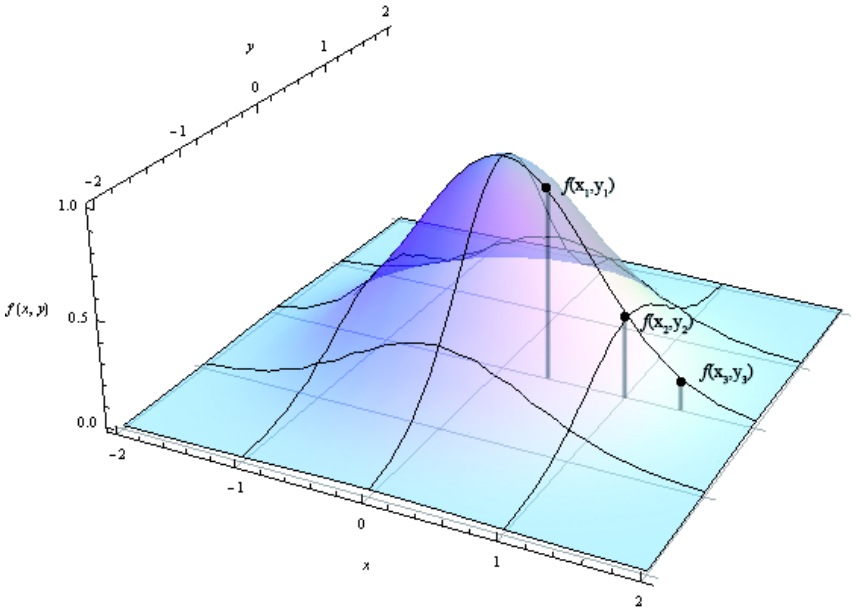

Figure 2: Domination in a two-dimensional case

shrink the searching range over the control space with the ranking information of some grid points. As we showed in the two previous graphs, given the domination relation of any two grid points, all of the points in the "downward" direction beyond the lower ranking point can be skipped. This significantly saves the computational time for solving the Bellman equation.

2.3 Algorithm

Proposition 1 helps to shrink the searching range over the control space in both single and multi-dimensional cases. Here we give the algorithm of the RGS method for a one-dimensional and a two-dimensional case, which applies the proposition above. A similar algorithm can also be applied in higher dimensional DP problems. For the purposes of comparison, this algorithm is based on the bracketing technique as in İmrohoroğlu, İmrohoroğlu and Joines (1993).

2.3.1 Algorithm 1: one-dimensional case

For a Bellman equation with one-dimensional control space C , given the state variable a_t, we have the following algorithm:

Step 1: Set the maximum iteration number k according to the precision required.

Step 2: Discretize the control space C in a closed subset $[c_{\min}, c_{\max}]$. Five grid

points are evenly distributed in the search space. Label the five points with index from lowest to highest as $\{x_A, x_B, ..., x_E\}$ (see Figure 3).

Step 3: Evaluate the value of point A and B as $V(x_A)$ and $V(x_B)$.

Step 4: If $V(x_A) > V(x_B)$, go back to step 2 and reset the searching space as $[x_A, x_B]$. If not, compute $V(x_C)$. Next, if $V(x_B) > V(x_C)$, go back to step 2 to reset the searching space as $[x_A, x_C]$. If not, compute $V(x_D)$. Next, if $V(x_C) > V(x_D)$, go back to step 2 and reset the searching space as $[x_B, x_D]$. If not, compute $V(x_E)$. Next, if $V(x_D) > V(x_E)$, go back to step 2 to reset the searching space as $[x_C, x_E]$. If not, $V(x_E)$ is the highest among the five grid points, go back to step 2 and reset the searching space as $[x_D, x_E]$. The iteration number increases by one.

Step 5: Keep going until the maximum iteration is reached. The point that dominates in the last iteration is the numerical solution of the Bellman equation.

Notice that in the best case of using the RGS method, we only need to evaluate and compare two grid points in each iteration, which is the case when $V(x_A) > V(x_B)$; in the grid search method employed by İmrohoroğlu, İmrohoroğlu and Joines (1993), one has to go over every grid point to find the optimum. By using the local information of the Bellman equation, RGS skips evaluating and comparing unnecessary grid points and hence speeds up the search in each iteration.

2.3.2 Algorithm 2: two-dimensional case

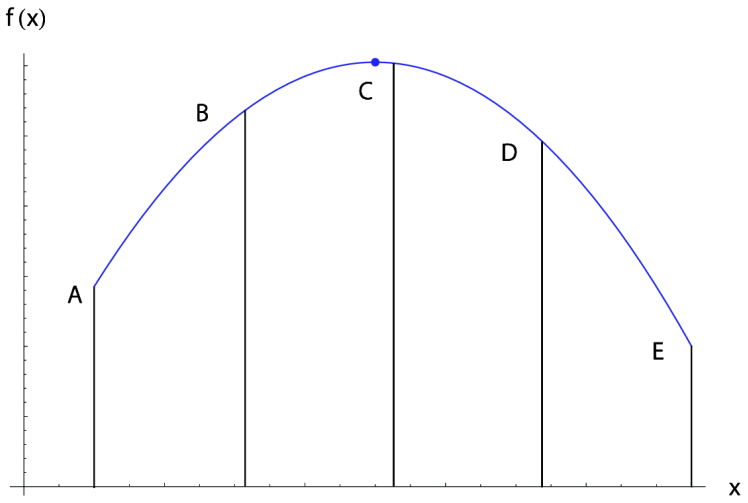

Figure 3: Rapid grid search in a one-dimensional case

For a Bellman equation with two-dimensional control space C , given the state variable a_t, we can apply the following algorithm:

Step 1: Set the maximum iteration number k according to the precision required.

Step 2: Discretize the control space C in a closed subset. 25 grid points, five on each dimension, are evenly distributed in the search space. Label the 25 points with index from lowest to highest as $\{x_1, x_2,..., x_{25}\}$ (see Figure 4).

Step 3: Evaluate and compare $V(x_1)$, $V(x_2)$, $V(x_3)$, $V(x_4)$ and $V(x_5)$ by using Algorithm 1 in the one-dimensional case. Obtain the maximum from these five points. Then move to x_6, x_7, x_8, x_9, and x_{10}. Find the maximum again from these five points by using the RGS Algorithm 1. Keep going for the remaining points. We

end up with five local maximum points for each round. We then compare these five points to find the global maximum. Shrink the search space to the neighborhood around this global maximum point and go back to step 2. The iteration number increases by one.

Step 4: Keep going until the maximum iteration is reached. The point that dominates in the last iteration is the numerical solution of the Bellman equation.

Notice that for the sake of simplicity, Algorithm 2 is a straightforward extension of Algorithm 1 and it does not apply the RGS in its full length. For example, we could further apply the RGS in step 3 when evaluating and comparing the five local maximum points to determine the global maximum. There are different ways to improve the efficiency here, and we would like to leave that to readers to deal with their specific problems. However, it is worth noting that in the best case of each iteration, we only need to go over four grid points out of 25 by using the RGS. For example, if we evaluate points 1, 2, 6 and 7 and we have $V(x_1) > V(x_2) > V(x_6) > V(x_7)$ or $V(x_1) > V(x_6) > V(x_2) > V(x_7)$, we do not need to continue searching other points because $V(x_1)$ is the global maximum out of these 25 points.

3 Application

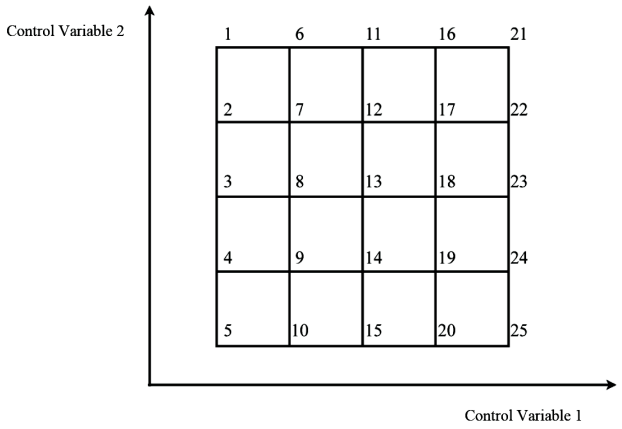

Figure 4: Rapid grid search in a two-dimensional case

This section describes the application of our algorithm to two DP problems in macroeconomics. In both of the problems, it is clearly shown that the RGS method is significantly more efficient than the benchmark grid search method–the bracketing algorithm in terms of computation speed.

3.1 One-dimensional Model

İmrohoroğlu, İmrohoroğlu and Joines (1995, 1999) study the optimal social security replacement rate and the welfare benefits associated with it in an overlapping generations general equilibrium framework. Individuals face mortality risk and idio-

syncratic income shock over the life cycle. However, due to the absence of a private credit annuity market, they have to use savings to self-insure against these shocks. In the model, each individual has to solve a finite-horizon finite-state DP problem, which is summarized in the following Bellman equation

$$V_j(a,s) = \max_{(c,a')\in\Omega_j(a,s)} \{U(c) + \beta\psi_{j+1}E_{s'}V_{j+1}(a',s')\}, j = 1,2,...,J \tag{2}$$

where j is the age, a is the asset holding at the beginning of age-j, s is the state associated with employment status, c represents consumption, and ψ_{j+1} stands for the conditional probability of survival from age j to age $j+1$. $\Omega_j(a,s)$ is the budget constraint for age j. Notice that we can use the budget constraint to reduce the control variable to be the only asset holding for next period a'. The state and control spaces thus coincide for this example.

As in İmrohoroğlu, İmrohoroğlu and Joines (1993), solving the Bellman equation above involves a grid search based on a bracketing technique that we use as a benchmark case.[3] We then use the RGS method to repeat the exercise. In both cases, to solve the Bellman equation, first, we discretize the control space $C = [0, 40]$ by 4097 equally distributed grid points. The same grid points are also used for the control

[3]The Fortran code to compute the model is downloaded from http://dge.repec.org/codes/marimon-scott/Imrohoroglu/.

variable a'. The total number of theoretical grid points is $4097 \times 4097 \times 2 \times 44$ (working age) + $4097 \times 4097 \times 21$(retirement age) = $1.82961e+9$ in each iteration. The model converges to the tolerance of 10^{-3} after 7 iterations for both methods.[4] As shown in Table 1, in the benchmark experiment, it takes 9.26 seconds under the current hardware.[5] Using our Algorithm 1 above, the computing time is reduced to 5.09 seconds. The RGS method saves 45.03% of computing time.[6] The time efficiency comes from the fact that the RGS method can skip lots of unnecessary grid points. Again, as shown in Table 1, the benchmark bracketing method, although already a huge gain from the brute force grid search, still needs to search 1.48898e+8 grid points totally during 7 iterations; while the RGS method further shrinks the number of grid points searched to $6.26218e+7$, which is only 42.06% of the gird points searched by the benchmark case. In other words, the RGS method speeds up the computation by skipping 57.94% of grid points searched by the standard bracketing method.[7]

As a robustness check, we also double the number of grid points on the state and control spaces to 8193 and solve the model. The model again converges to the tolerance of 10^{-3} after 7 iterations for both methods. Table 1 shows that the

[4]Total number of grid points that need to be searched using a brute force grid search is 1.28073e+10.

[5]Environment: AMD Athlon×2 5200, 4G RAM, Intel Fortran compiler for Linux.

[6]Time efficiency is defined as 1 - elapsed time$_{RGS}$/elapsed time$_{benchmark}$.

[7]Searching efficiency is defined as 1 - grids number searched$_{RGS}$/grids number searched$_{benchmark}$.

	Benchmark method		**RGS method**	
Theoretical grid number	1.28073e+10	5.12166e+10	1.28073e+10	5.12166e+10
Grid number searched	1.48898e+8	3.28655e+8	6.26218e+7	1.36070e+8
Searching efficiency	—	—	57.94%	58.60%
Elapsed time (seconds)	9.26	20.87	5.09	11.98
Time efficiency	—	—	45.03%	42.58%

Table 1: Results for RGS method, one-dimensional case

searching efficiency and time efficiency are very close to the case with 4097 grid points.

3.2 Two-dimensional Model

Braun and Nakajima (2009) investigate an infinite-horizon endogenous growth model with human capital and Epstein-Zin preference. In their model, an individual solves the following DP problem

$$V(a_t) = \max_{c_t, a'_t, \omega_{k,i}} \{c_t^{1-1/\psi} + \beta(V(a_t'^{1-\gamma}))^{\frac{1-1/\psi}{1-\gamma}}\}^{1/(1-1/\psi)} \tag{3}$$

subject to

$$a' = (a - c)\{R'_k \omega_k + R'_h(1 - \omega_k)\}$$

where a is the asset holding at the beginning of the period, a' is the asset holding for the next period, c is consumption, R'_k and R'_h are the returns to physical and

human capital for the next period, respectively, and ω_k is the share of physical capital in total capital.[8] Replacing c in the utility function by the budget constraint, we can reduce this DP problem to a two-control-variable Bellman equation. The agent chooses a' and ω_k to maximize the Bellman equation. In addition to providing a two-dimensional example for solving DP problems, the specification of the model allows nearly closed-form solutions, which offers a nice test case to check the accuracy of our numerical algorithm.

To solve this infinite-horizon DP problem, we first discretize the state space by 1000 grid points and the control space by 4097 grid points in each dimension. The total number of theoretical grid points is $4097 \times 4097 \times 1000$ in each iteration. Then we repeatedly solve the Bellman equation for each grid point on the state space until the value function converges to 10^{-8} tolerance and the solution precision reaches the range of 10^{-4}. We again use the bracketing method in İmrohoroğlu et al. (1993) as the benchmark method, and then use our RGS method as described in Algorithm 2 above to repeat the exercise. In both cases, the model converges to the tolerances after 11 iterations.[9] As shown in Table 2, a huge efficiency gain shows up both in the computing time and in the number of grids actually searched. The RGS method

[8] The original model in Braun and Nakajima (2009) allows an idiosyncratic uninsurable shock to the return on human capital. To simplify the computation time, we shut down this idiosyncratic shock in our computation. Our model thus is a deterministic version of Braun and Nakajima's original model.

[9] Total number of theoretical grids thus is $4097 \times 4097 \times 1000 \times 11 = 1.84639\text{e+}11$.

	Benchmark method		**RGS method**	
Theoretical grids number	1.84639e+11	7.38378e+11	1.84639e+11	7.38378e+11
Grid number searched	2.75000e+6	3.02500e+6	1.12868e+6	2.12297e+6
Searching efficiency	—	—	58.96%	29.82%
Elapsed time (seconds)	62.21	68.18	21.24	38.18
Time efficiency	—	—	65.86%	44.00%

Table 2: Results for RGS method, two-dimensional case

saves about 66% of computing time and about 59% of grid points searched. As a robustness check, we then double the number of grid points in the control space from 4097 to 8193 and recompute the model. Not surprisingly, significant efficiency gains appear again with finer grids using the RGS method.

4 Conclusion

We introduce a rapid grid search method in solving the dynamic programming problems in economics,which inherits the advantages of the standard grid search method. Going one step further, by using local information of the Bellman equation, this method can significantly increase the efficiency in solving DP problems by reducing the grid points searched in the control space. By applying this method to a one-dimensional and a two-dimensional case, respectively, we obtain a significant gain in efficiency by reducing the computational time compared to the benchmark grid search algorithm. This method can also be easily implemented and applied to

higher dimensional DP problems. Therefore, it can offer a possible way to help relieve the "curse of dimensionality" arising from the high-dimensional DP problems in economics.

References

[1] Braun,A. and T. Nakajima (2009): "How Large Is the Intertemporal Elasticity of Substitution?" unpublished mimeo.

[2] Grune, L. and W. Semmler (2004) : "Using Dynamic Programming with Adaptive Grid Scheme for Optimal Control Problems in Economics," *Journal of Economic Dynamics and Control* 28, 2427 – 2456.

[3] İmrohoroğlu,A., S. İmrohoroğlu and D. Joines (1993): "A Numerical Algorithm for Solving Models with Incomplete Markets," *International Journal of Supercomputer Applications and High Performance Computing* 7, 211-230.

[4] İmrohoroğlu,A., S. İmrohoroğlu and D. Joines (1995): "A Life Cycle Analysis of Social Security," *Economic Theory* 6, 83-114.

[5] İmrohoroğlu, A., S. İmrohoroğlu and D. H. Joines (1999): "Social Security in an Overlapping Generations Model with Land," *Review of Economic Dynamics* 2, 638-665.

[6] Stokey, N., R. Lucas and E. C. Prescott (1989): *Recursive Methods in Economic Dynamics*, Harvard University Press.